|  |
| --- |
| **Teacher: ROSE** |
| **Date: 8/14/18** |
| **Subject / grade level: Computer Science - AP CS Principles (9-12)** |
| **Materials:**  JSONdemo\_teacher  JSONdemo\_student |
| **Essential Question(s):**  What is a JSON object and how can we parse its data?  How are networking packets structured/layered? |
| **Essential Standards (NGSS) and (CCSS):**   |  |  |  | | --- | --- | --- | | **Science & Engineering Practices (SEPs)** | **Disciplinary Core Ideas (DCIs)** | **Crosscutting Concepts (CCs)** | | NA | NA | **NA** |   **CSTA Standards:**  **Computing Systems**  3A-CS-01: Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.    3A-CS-02: Compare levels of abstraction and interactions between application software, system software, and hardware layers.  **Networks and the Internet**  3A-NI-04: Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.  **Data and Analysis**  3A-DA-09: Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.  3A-DA-10: Evaluate the tradeoffs in how data elements are organized and where data is stored. |
| **Lesson objective(s) - *Students will demonstrate understanding or learning around the following Big Ideas:***  *SWBAT read, write, and parse JavaScript Object Notation. JSON is a lightweight data-interchange format. It is easy for humans to read and write and easy for machines to parse and generate. It is composed of objects (collections of key-value pairs) and lists (arrays). In order to parse JSON, students must be comfortable manipulating objects and arrays.* |
| **Differentiation strategies to meet diverse learner needs:**  **NA** |
| **ENGAGEMENT (*Anchoring Phenomenon*)**   * Teacher will introduce students to Wireshark and capture live data of their WiFi activity, showing the resulting PCAP file (hex). * Students will probably have a multitude of questions about the data generated relating to networking and packets, but the big questions overall * How is the data encoded? * How can we make sense of all of this data? * Can we visualize this data? |
| **EXPLORATION**   * *Teacher will then export the same data in JSON format and share the file with students.* * *Students may also choose to generate their own JSON file from Wireshark* * *Students can open the file and copy/paste the JSON data in a variable in the web console* * *How is the data structured? Try using brackets [] and the dot (.) operator to access the various elements* * *A) The data we care about lies within the layers object (frame,Ethernet,IP, TCP/IP)* * *B) Once a particular layer is accessed, brackets [] must be used to access its properties, why?* * Draw a hierarchical diagram of the data as a class |
| **EXPLANATION**   * *Why do you think it’s called JavaScript Object Notation?* * *ie What does it relate to that we’ve learned about in JavaScript already? (objects, arrays, variables)* * *See json.org for definitions and examples* * *Shiffman videos: What is JSON* [*Part 1*](https://www.youtube.com/watch?v=_NFkzw6oFtQ)[*Part 2*](https://www.youtube.com/watch?v=118sDpLOClw) |
| **ELABORATION**   * [*Jsonlint.com*](http://jsonlint.com/) *– students will be given example JSON and asked to determine if it is valid. This can be verified online with any json linter.* * *Students will be given sample data sets and asked to generate the corresponding JSON to store it* * *Samples may be taken from online data/APIs that students use on a daily basis* |
| **EVALUATION**  Teacher will show output from jsonDemo\_teacher and then ask student to complete the tasks in jsonDemo\_student:   * *//look at each packet i in data* * *//create a layer variable* * *//store layers in a string variable for later use* * *//2.1: Print layer information* * *//2.2: Print layer 1 frame length* * *//2.3: Print layer 2 ethernet mac addresses: source --> destination* * *//2.4: Print layer 3 IP addresses: source --> destination (only if present)* * *//2.5: Print out 4 TCP/UDP port numbers: source --> destination (only if IP layer exists* |
|  |